Lívia Barbosa

Plano de Testes - Verificação da API ServeRest

1. Apresentação

O planejamento de testes em questão descreve os meios e estratégias a serem seguidas para validar a qualidade da API ServeRest, a qual simula uma loja virtual com funcionalidades de gerenciamento de usuários e itens, bem como simulação de compras de produtos.

A versão da API testada é a 2.29.7, a qual corresponde à versão mais atual da aplicação, podendo ser acessada por meio do seguinte link: https://compassuol.serverest.dev/.

2. Objetivo

Identificar falhas funcionais na API ServeRest por meio de testes funcionais realizados na ferramenta Postman, utilizando diferentes técnicas para guiar a execução dos testes, a fim de propor melhorias para as funcionalidades testadas e apontar defeitos que possam comprometer a qualidade dos serviços.

3. Escopo

Serão aplicados testes funcionais em todas as funcionalidades da aplicação, abrangendo os serviços de login, gerenciamento de usuários, gerenciamento de produtos e gerenciamento de carrinhos. Entretanto, é relevante citar que a execução dos testes contemplará prioritariamente os serviços considerados mais críticos.

Não serão realizados testes não-funcionais, testes de caixa branca e testes end-to-end.

4. Pessoas envolvidas

Papel	Nome	Responsabilidade
Testador	Lívia	Executa e documenta os testes realizados nos serviços de login e de gerenciamento de usuários, produtos e carrinhos.

5. Análise

A API submetida a teste é uma aplicação RESTful, a qual possui uma arquitetura baseada em endpoints que utilizam os métodos HTTP padrão (POST, GET, PUT e DELETE). Parte dos endpoints requer autenticação via token Bearer, cuja validade é de 10 minutos a partir da autenticação do usuário. Outrossim, os dados trafegam em formato JSON, com regras de negócio importantes para validação de campos obrigatórios, formatação dos dados e consistência nas entradas.

É relevante destacar que as funcionalidades estão interligadas, exigindo dados e execuções prévias para testar determinados módulos, tais como:

- Cadastrar um usuário para testar o módulo de login;
- Cadastrar um usuário administrador para testar funcionalidades exclusivas do administrador, como gerenciamento de usuários e produtos;
- Cadastrar previamente um produto e um usuário para testar o módulo de carrinhos.

Inicialmente, foram identificados riscos relacionados a permissões administrativas, os quais podem impactar diretamente a integridade dos dados dos usuários e produtos contidos no sistema.

Em suma, com base nessa análise inicial e em uma análise detalhada posterior, é possível observar que os módulos mais críticos estão concentrados nos módulos de gerenciamento de produtos, gerenciamento de usuários e gerenciamento de carrinho, respectivamente, visto que estes possuem mais riscos vinculados, como apresentado na Seção 14.

6. Técnicas de teste utilizadas

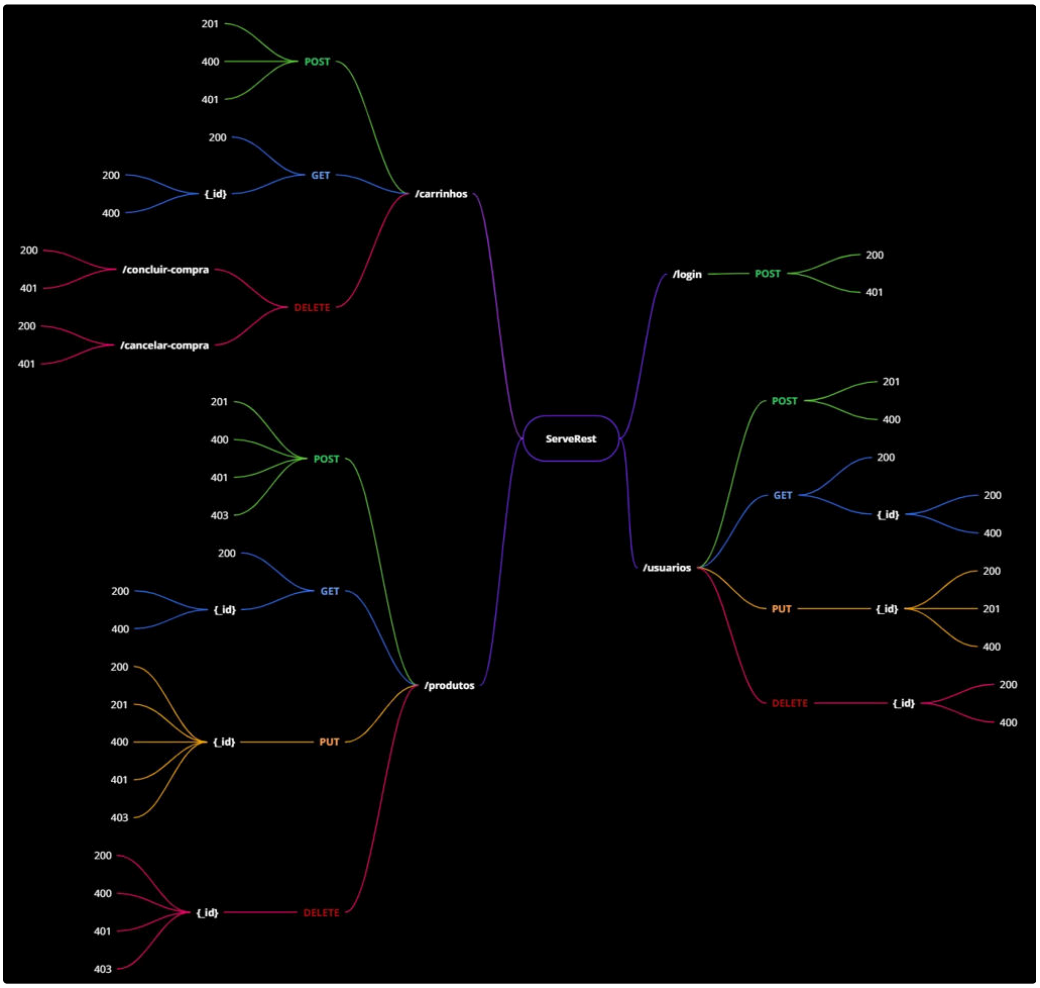
Serão aplicadas técnicas de caixa preta em todos os módulos da API, sendo elas:

- Partição de Equivalência, para validar diferentes grupos de entradas válidas e inválidas nos campos;
- Valor Limite de 3 pontos, para verificar se os campos de senha possuem um tratamento de acordo com as regras de negócio;
- Null, Zero, Vazio, para verificar o comportamento da API em cenários de dados obrigatórios nulos ou em branco;
- Tabela de Decisão, para validar diferentes combinações de regras de negócio ao realizar o login.

Todas as técnicas serão apoiadas pela Heurística CRUD, uma vez que esta garante a cobertura de cenários de criação, leitura, atualização e exclusão de dados.

7. Mapa mental da aplicação

Para compreender a fundo o funcionamento da API, juntamente com os seus serviços, foi desenvolvido um mapa mental com todos os módulos da aplicação incluídos.



8. Módulos a serem Testados

- Login;
- Usuários;
- Produtos;
- Carrinhos.

9. Local dos Testes

- **Ambiente:** Homologação
- **URL de Acesso:** https://compassuol.serverest.dev/
- **Dados de Acesso:**
  - **Usuário não administrador:**
    - **Email:** usuarioteste@teste.com

        - **Senha:** teste
    - **Usuário administrador:**
        - **Email:** usuarioadm@teste.com
        - **Senha:** teste

10. Recursos necessários

- Máquina com acesso à internet;
- Navegador Google Chrome;
- Plataforma para reportar os bugs e melhorias (Jira);
- Plataforma para criar e executar os testes (Postman);

11. Critérios usados

- **Critérios de Início:**
    - Versão da API estável no ambiente de homologação;
    - Acesso às Histórias de Usuário da aplicação;
    - Criação da Coleção no Postman.
- **Estratégia de Teste:**
    - Baseada em Riscos.
- **Critérios de Conclusão:**
    - Todos os casos de teste executados;
    - Relatório de issues e melhorias finalizado.

12. Cenários de teste

Todos os cenários de teste podem ser acessados por meio do seguinte documento: Cenários de teste.

13. Priorização da execução dos cenários

Os cenários de teste foram priorizados de acordo com o nível de risco das funcionalidades, bem como pela relevância dentro da lógica da aplicação, ou seja, funcionalidades consideradas com alta classificação de risco para o funcionamento da API e com frequência de uso pelos usuários. Dessa forma, foram considerados cenários de teste que abrangem a lógica básica do sistema, como também aqueles relacionados à autenticação do usuário, fundamental para o controle de acesso às diferentes funcionalidades.

Por fim, o resultado da priorização dos cenários de teste foi o seguinte:

Alta prioridade: Cenários do módulo de login → CT001, CT002, CT003, CT004; Cenários do módulo de produto → CT033, CT034, CT035, CT036, CT037, CT038, CT039, CT040,

CT041, CT042, CT043, CT044, CT052, CT053, CT054, CT055, CT056, CT057; Cenários de módulo de usuários → CT007, CT015, CT025; Cenários do módulo de carrinhos → CT061, CT062, CT063, CT068

Média prioridade: Cenários de módulo de usuários → CT005, CT006, CT008, CT009, CT010, CT011, CT012, CT013, CT014, CT016, CT017, CT018, CT019, CT020, CT021, CT022, CT029, CT030, CT031, CT032; Cenários do módulo de carrinhos → CT058, CT059, CT060, CT064.

Baixa prioridade: Cenários do módulo de produto → CT045, CT046, CT047, CT048, CT049, CT050, CT051; Cenários de módulo de usuários → CT023, CT024, CT026, CT027, CT028; Cenários do módulo de carrinhos → CT065, CT066, CT067, CT069, CT070, CT071.

14. Matriz de risco

Desenvolveu-se uma matriz de risco com base em todos os possíveis riscos das funcionalidades da API, onde:

- **Pontuação de Probabilidade:** de 1 a 5, onde:
  - 1 = Muito baixa;
  - 2 = Baixa;
  - 3 = Média;
  - 4 = Alta;
  - 5 = Muito alta.
- **Pontuação de Impacto:** de 1 a 5, onde:
  - 1 = Muito baixo;
  - 2 = Baixo;
  - 3 = Médio;
  - 4 = Alto;
  - 5 = Muito alto.
- **Pontuação de Classificação:** resultado da multiplicação entre a pontuação de probabilidade e a pontuação de risco.

A Matriz de Risco pode ser acessada pelo link a seguir: Matriz de Risco.

15. Cobertura dos testes

15.1. Cobertura de Entrada

15.1.1. Path Coverage

Analisa a cobertura da suíte de testes de acordo com os endpoints que a API possui.

A cobertura atual dos testes é de **78%**, com testes para 7 dos 9 endpoints.

15.1.2. Operator Coverage

Confere a cobertura de testes de todos os métodos existentes na API REST.

A cobertura atual dos testes é de **62,5%**, com testes para 10 dos 16 métodos.

15.1.3. Content-Type Coverage

Analisa a cobertura de testes automatizados onde o content-type está sendo exibido em cada endpoint de entrada dos métodos.

A cobertura atual dos testes é de **100%**, cobrindo todos os 6 content-type de entrada dos métodos.

15.2. Cobertura de Saída

15.2.1. Status Code Coverage

Avalia quais status codes existentes em cada endpoint estão cobertos pelos testes.

A cobertura atual dos testes é de **68%**, cobrindo 26 dos 38 status code dos métodos.

16. Testes candidatos à automação

Grande parte dos cenários de teste foram considerados candidatos à automação, visto que possuem entradas e saídas objetivas, juntamente com resultados previsíveis. Entretanto, em cenários de teste que contemplam regras de negócio que exigem a conexão entre diferentes endpoints da API - como a impossibilidade de excluir um usuário ou produto que esteja vinculado a um carrinho -, a automatização se torna muito complexa, visto que diversas operações são executadas por diferentes usuários e, por conseguinte, se torna difícil mensurar o estado final do sistema corresponde exatamente ao esperado sem a intervenção e validação manual.

17. Como os resultados de teste serão divulgados

Os resultados serão divulgados por intermédio de uma coleção gerada pela ferramenta Postman, contendo os cenários de teste mapeados, além de um documento com o mapeamento de issues e melhorias apontadas com base nos testes realizados. Ademais, o

planejamento e os resultados serão transmitidos por meio de uma apresentação verbal em uma reunião com a equipe.

18. Cronograma

ATIVIDADES	11/08	12/08	13/08	14/08	15/08
Produção do plano de testes					
Criação dos scripts de teste					
Execução dos testes					
Interpretação dos resultados					
Produção do relatório de issues e melhorias					
Apresentação dos resultados					

19. Mapeamento de Issues e Melhorias

O registro das issues e propostas de melhorias para o funcionamento da API pode ser acessado por meio do seguinte link:

Mapeamento de Issues e Melhorias.